6-2011

BREP Identification During Voxel-Based Collision Detection for Haptic Manual Assembly

Daniela Faas
Iowa State University

Judy M. Vance
Iowa State University, jmvance@iastate.edu

Proceedings of the ASME 2011 World Conference on Innovative Virtual Reality
WINVR2011
June 27-29, 2011, Milan, Italy

WINVR2011-5524

BREP IDENTIFICATION DURING VOXEL-BASED COLLISION DETECTION FOR HAPTIC MANUAL ASSEMBLY

Daniela Faas
Department of Mechanical Engineering
Virtual Reality Applications Center
Iowa State University
Ames Iowa 50011
dfaas@iastate.edu

Judy M. Vance
Department of Mechanical Engineering
Virtual Reality Applications Center
Iowa State University
Ames Iowa 50011
jmvance@iastate.edu

ABSTRACT

This paper presents a novel method to tie geometric boundary representation (BREP) to voxel-based collision detection for use in haptic manual assembly simulation. Virtual Reality, in particular haptics, has been applied with promising results to improve preliminary product design, assembly prototyping and maintenance operations. However, current methodologies do not provide support for low clearance assembly tasks, reducing the applicability of haptics to a small subset of potential situations. This paper discusses a new approach, which combines highly accurate CAD geometry (boundary representation) with voxel models to support a hybrid method involving both geometric constraint enforcement and voxel-based collision detection to provide stable haptic force feedback. With the methods presented here, BREP data can be accessed during voxel-based collision detection. This information can be used for constraint recognition and lead to constraint-guidance during the assembly process.

Keywords: Virtual Assembly, Virtual Reality, Human Computer Interaction, Haptic Feedback, Mechanical Design.

INTRODUCTION

Assembly operations rely on accurate physical simulation to provide the user with realistic feedback to evaluate the design. Haptic force feedback has been shown to improve the sense of presence of the operator while in a virtual environment [1]. However, low clearance assembly is still difficult because most methods of collision detection and force calculations rely on approximated CAD models in order to maintain haptic refresh rates of at least 1000 Hz. One approach to solving these issues is to combine voxel-based collision detection and geometric constraint recognition in order to provide a system where users can interact with CAD models naturally and intuitively during low clearance assembly. Voxel-based collision detection allows fast and reliable force calculations, but encounters difficulties simulating low clearance assembly because voxel methods approximate the surface of the CAD model. This approximation is dependent on the voxel size. Small voxels are required in low clearance assembly situations; however, the number of voxels in the scene is limited by the computational power and storage capacity of the computer.

Methods using geometric constraint recognition also support simulated contact between CAD models, but these methods face challenges in modeling force interactions. Often, these methods are implemented as "snap-to" scenarios where parts snap to final assembly position when in contact. A particular problem with these methods is the need to pre-define the geometric constraints prior to starting the simulation. Depending on the complexity of the assembly, the manual pre-processing can add significant effort to the task of preparing the models for the virtual environment.

A promising approach is to combine voxelized models with BREP models to provide natural interaction with parts. The key to implementing such a method is the need to relate geometric information to the voxel-based representation. The method described in this paper binds BREP data to the voxel model without pre-processing and can be used to determine which BREPs are colliding at haptic refresh rates. The move between voxel-based collision detection and force calculation and geometric constraint modeling is now possible.

BACKGROUND

This section reviews and summarizes several approaches for performing virtual assembly simulations. Most of the early assembly methods did not use collision detection and depended on pre-defined assembly positions of parts, which allowed them to be snapped into place when they were close to their final position. Using geometric constraints allowed for precise placement of parts, but these constraints were typically pre-defined and reduced the opportunity for interactive changes in the assembly sequence while in the virtual environment.

1. Physics-Based Assembly Simulation

This first category consists of systems and applications that use physical properties to simulate the interaction of parts with the environment. These applications typically allow the user to move parts freely in the environment. Examples of physics-based simulations include VSHOP by Pere *et* al. [2], which used collision detection based on boundary boxes to avoid part interpenetration.

VEDA (Virtual Environment for Design for Assembly) by Gupta *et* al. simulated part trajectories once collision occurred [3-5]. However, the assembly process was limited to two-dimensional (2D) models. HIDRA (Haptic Integrated Dis/Re-assembly Analysis), developed by Coutee *et* al., allowed the user to grab models between two fingertips, but had only limited 3D manipulation ability [6, 7].

The Virtual Environment for General Assembly (VEGAS), implemented physics-based modeling for parts using VPS (Voxmap PointShell™) [8]. Kim and Vance [9] then developed NHE (Network Haptic Environment), which enable assembly tasks to be evaluated by individuals in geographically distinct locations, each with a haptic device (Fig. 1).

Figure 1: VEGAS shown in a CAVE Environment [8]

2. Positional-Based Assembly Simulation

The second category of applications uses positional constraints to place parts within the virtual environment Positional-based assemblies do not allow for decision making during the assembly process. Positional-based methods require pre-processing of the final assembly position. Parts snap to final, pre-defined positions. One of the first applications that integrated positional constraint-based modeling was VADE (Virtual Assembly Design Environment), developed in 1995 by Jayaram *et* al. [10]. Later versions of VADE included improvements on this approach.

3. Geometric Constraint-Based Assembly Simulation

Geometric constraints can be used to place parts precisely. In the case of a pin inserted into a hole, the alignment of the pin axis with the hole axis is the geometric constraint related to the assembly process. Examples of applications that import geometric constraints directly from ProEngineer into the virtual environment are VADE [11], VECA (Virtual Environment for Collaborative Assembly) [12] and MIVAS (Multi-Modal Immersive Virtual Assembly System) [13] .

Hybrid approaches use a combination of physics and geometric constraint-based interactions to manipulate parts in a virtual environment. Loic *et* al. [14] developed a method that uses non-smooth contact dynamics to manipulate objects in the environment and render haptic forces. This method uses "guide planes" as visual cues and requires manual pre-processing. The geometric constraint is engaged when the moving part approaches the stationary part. Once the geometric constraint is engaged, collision detection and haptic rendering are suspended.

Marcelino *et* al. [15] developed a geometric constraint manager to simulate assembly and disassembly tasks in VR using direct interaction, automatic geometric constraint recognition, geometric constraint satisfaction and constrained motion. Seth *et* al. [16] developed a feature-based approach to geometric constraint recognition by taking advantage of dynamically contacting BREP features to predict assembly intent. This system did not support haptic force feedback, but used Phantom Omnis for selecting objects (Fig. 2).

Figure 2: Dual-Handed Assembly Simulation [16]

Iacob *et* al. [17, 18] proposed a new method to manage collisions for contact identification. Polyhedron and kinematic constraints are generated and contact information is created. The kinematic constraints are used to remove the collision detection between those contacts, therefore allowing assembly and disassembly to occur.

There are many more examples to provide support for assembly scenarios in VR. All of them incorporate some type of human-computer interaction (HCI) device like a glove or haptic feedback. For a more comprehensive summary of previous research efforts, see [19, 20].

4. Geometric Reasoning for Assembly

Other efforts in assembly simulation include automatic path generation or using algorithms to allow (automated) geometric reasoning about assemblies. Most of these constraints only consider geometric constraints arising from the assembly and do not account for issues related to the manipulation system (human, robot, etc.) [21]. An automatic disassembly planner can provide information on the precedence of removals for parts. Path planners for disassembly generate the sequence of part removal required to extract a certain part from an assembly [22]. One important requirement is the capability to demonstrate that all products can be serviced during the operations. While these approaches are beneficial to assembly planning, they remove human decision making from the actual assembly or disassembly.

VOXMAP POINTSHELL METHOD

The research described here results in a mapping of b-rep entities to voxels in a 3D CAD model. The voxels form the underlying representation for the voxmap-pointshell method of collision detection and six degree-of-freedom force modeling [23-25]. The Voxmap PointShell™ (VPS) software, licensed from Boeing, was used in the development of the research presented here. The voxmap-pointshell method can sustain a consistent 1000 Hz haptic refresh rate to render forces smoothly. Previous investigations have shown that VPS was highly suitable for providing haptic force feedback for assembly scenarios [26]. The next sections provide a short overview of the VPS method.

1. Voxelization Process

Tessellated data from the CAD model is used as input for the voxelization process. In the voxmap-pointshell method, models in the scene are voxelized in a pre-processing step to a voxel size specified by the user. To begin voxelization, the model is divided into regions of free space, object surface, and object interior. This space is partitioned using a volume occupancy map or voxmap to create the voxels. Static objects are represented by voxels, while dynamic objects are represented by a pointshell set based on the voxelized model (Fig. 3).

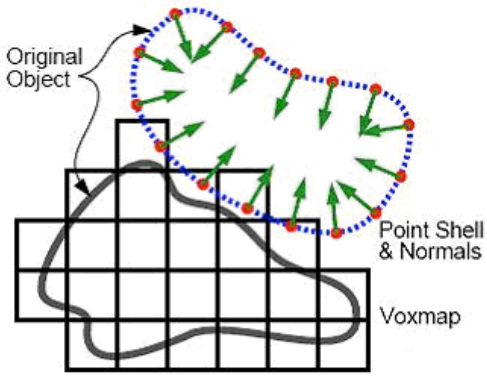

Figure 3: Static and dynamic CAD model representations [24]

The pointshell of the dynamic, or moving, object is created by identifying all of the center points of all surface voxels of the dynamic object. Surface normals are also a part of the pointshell data. Additional information can be stored as part of each voxel's data. Highly complex CAD models are modeled as a collection of small cubes, creating an approximated model. While this approximation is sufficient for most assembly clearance situations, it is not very suitable for low clearance scenarios.

2. Collision Detection

Extended discussions of the collision forces and torques calculated using the voxmap point shell method can be found in multiple papers and will not be reiterated here [23-25]. In short, collision detection using the voxmap-pointshell method is based on penetration of the pointshell into the set of voxels in the scene. When penetration occurs, a local force model is applied. The depth of interpenetration, *d*, is calculated as the distance the point penetrates into the voxel relative to a plane that passes through the center of the voxel perpendicular to the pointshell normal direction (Fig. 4). Based on this depth of penetration, a force is calculated using a simple spring-force model. The collision force and torque between colliding objects are proportional to the amount of inter-object penetration.

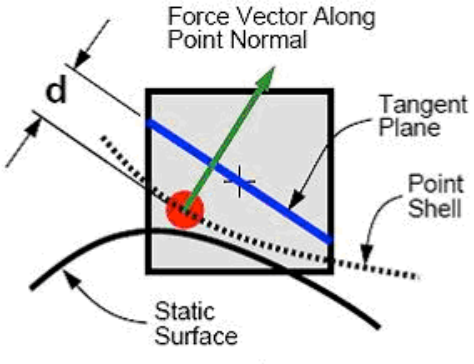

Figure 4: Tangent plane force model [24]

A coupling force and torque are applied between the virtual manipulator, or haptic handle (Fig. 5), and the dynamic object. The coupling force and torque consist of a linear spring-damper and a rotational spring-damper system. The primary purpose of the coupling force is to improve stability as the virtual dynamic object interacts with hard surfaces.

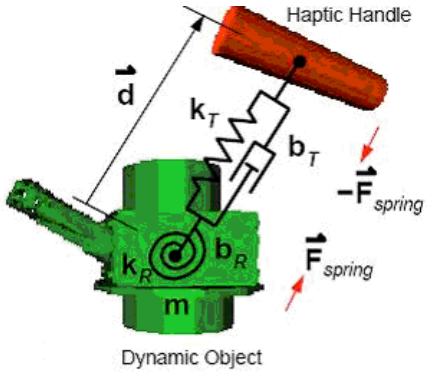

Figure 5: Virtual coupling model [24]

The total force is the summation of the coupling and collision forces. Rigid body dynamics are used to determine the state of the dynamic body at all times.

3. Improvements to the VPS Method

Several researchers have tried to improve upon the existing voxmap-pointshell method. Renz *et* al. [27] adapted the voxmap-pointshell method to provide a smoother surface representation. In addition, collision forces are varied to reduce "voxel noise". Other research focused on improving voxelization techniques to allow faster and more accurate voxel and pointshell models [28]. The voxelization algorithm navigates in the bounding box of each triangle of the original polygonal model detecting the probable surface-voxels. This approach prevents holes or excessive surface-voxels and generates voxels faster than the original method. Borro *et* al. [29] developed a method to optimize the voxel size for large virtual environments. Their method uses spatial partition techniques to improve the collision detection, but does not make any changes to the voxelized model. However, it is unclear if low clearance assemblies were achieved in either works.

BOUNDARY REPRESENTATION

There are several geometric representation schemes used in CAD systems. The main schemes are: wire frame representation, various surface modeling schemes, constructive solid geometry (CSG) and boundary representation solid modeling (BREP), as well as sweep representation. Based on each representation scheme, different information about the model is available from the CAD system (Fig. 6).

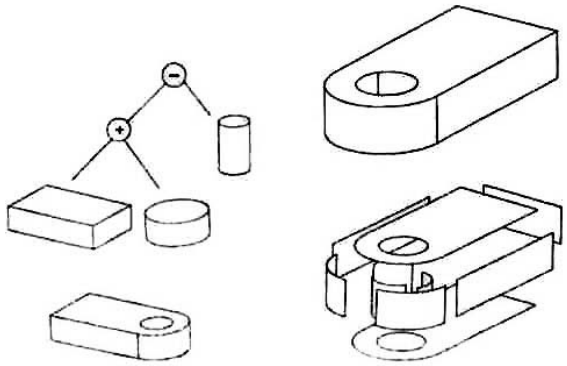

Figure 6: CSG (left) and BREP (right) entities in a generic CAD part [30]

The BREP is based on defining the limits of the object. A solid can be modeled as several connected surface elements. The main topological items are faces, edges and vertices. Consider the block and peg models in Figures 7 and 8. The block has six planar faces and one cylindrical face, whereas the peg has one cylindrical face and two planar faces. In addition, both models have two cylindrical edges.

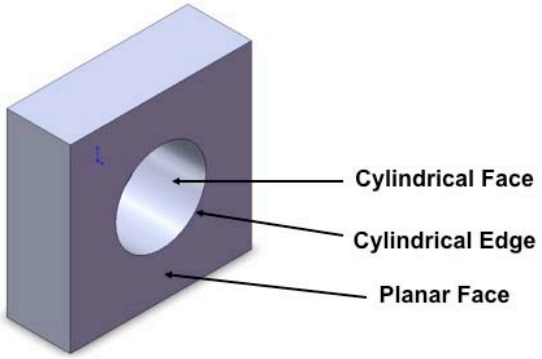

Figure 7: Geometric BREP entities in block (faces, edges and vertices)

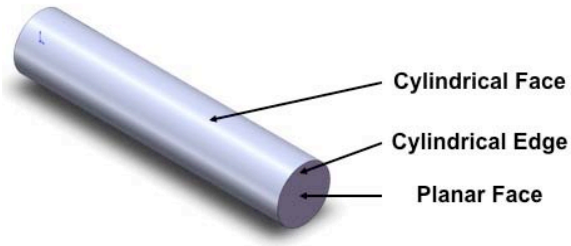

Figure 8: Geometric BREP entities in peg (faces, edges and vertices)

The advantage of BREP modeling is that the model has high spatial accuracy and motion constraints can be achieved by applying kinematic constraints. Because of their data structure, BREPs are used as the geometric basis in geometric constraint solvers. The goal of the geometric constraint solver

is to find all placements of the geometric entities that satisfy the given constraints. Geometric constraints are geometric entities such as distance, angle, parallel, perpendicular, concentric, and tangent.

While the underlying geometry representation in most CAD software is the BREP, for display and rendering purposes polygonal and triangular meshes are created from the BREPs. However, tessellated and BREP data are not necessarily associated with each other in data files that are easy to access from CAD programs. The research presented here results in a new method to bind BREP data to a voxel structure. Tying boundary representation to the voxels will allow the geometric constraint recognition algorithm to "piggy-back" on the voxmap pointshell collision detection method.

METHODOLOGY

In the hybrid method, three separate geometry models are needed for display, collision detection and force modeling: a tessellated model for visualization and display, a voxelized model for the voxmap pointshell method collision detection and force calculation, and a BREP model for geometric constraint recognition. (Fig. 9).

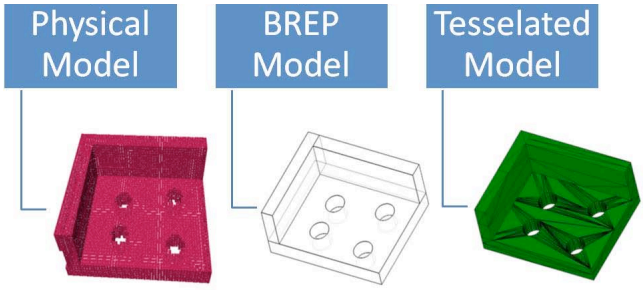

Figure 9: Separate data formats for use in VR environment

Fig. 10 shows the general process flow of the method discussed in this paper. This process occurs before the interactive simulation.

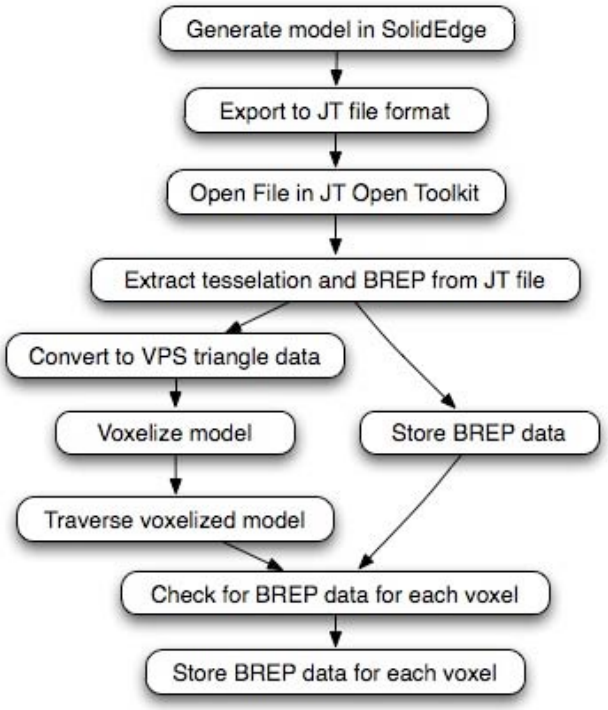

Figure 10: Process Flow

1. Data Extraction from JT

JT is a CAD neutral file format developed by Siemens PLM Inc. The CAD model can be generated with SolidEdge or any other CAD software that supports JT file export. The JT Open Toolkit is a C++ based API library, which contains functions to read and write JT files. JT Open Toolkit allows access to BREP and triangle data. The JT file contains BREP data along with the tessellation of the CAD part. This BREP data can be accessed and saved to a Parasolid (*.x_t) file. JT files can contain entire assemblies, subassemblies or individual parts (bodies).

Within those bodies, BREP and tessellated data is present (Fig. 11). Each BREP contains a transformation matrix and individual face and edge loop information with individual identifiers for each type of edge or face (can be geometric primitive or non-uniform rational basis spline). The JT file is read by the application using the *JtkCADImporter.h* method. The overall model structure is read by the method *JtkHierarchy.h* and the model tree is traversed in a top down approach by *JtkAssembly.h* at assembly and subassembly level and *JtkPart.h* at the body level (Fig. 11).

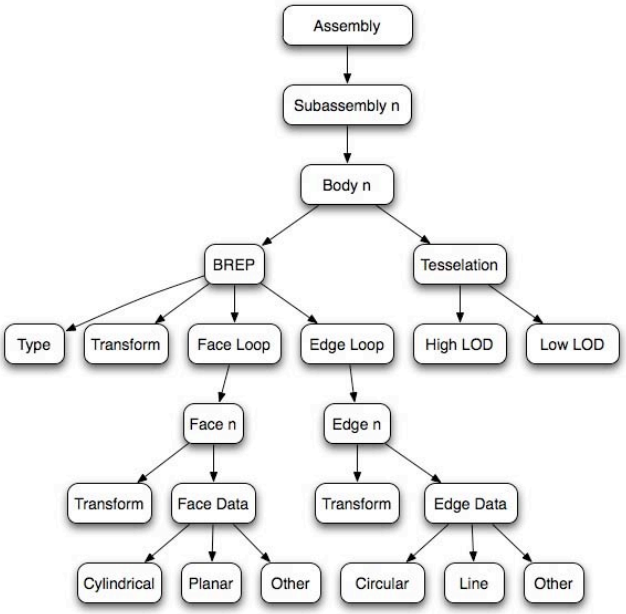

Figure 11: JT data structure

The querying of the tree at the part level can be done by *JtkShape.h*, which provides part information in terms of planar triangular strips. The triangulated data is available in high or low level of details (LOD). More level of details contains an increased number of triangles as compared to the low LOD. The JT file is queried using the *JtkShape.h* header functions at the part level, the algorithm scans one face at a time and then moves to the next face till all the faces are covered.

The face information is found out using the function *getInternal*. Each face contains vertex, texture and color information. In this research only the vertex information for each face is stored while the other information is ignored.

For each face, the vertex information is stored as a vertex array of triangular facets. The vertices of the triangular facets are ordered using the right hand rule. For example in Fig. 12, the planar face is divided into 3 triangles with vertices of 2-1-0, 4-3-2, 1-2-3 and 3-4-5.

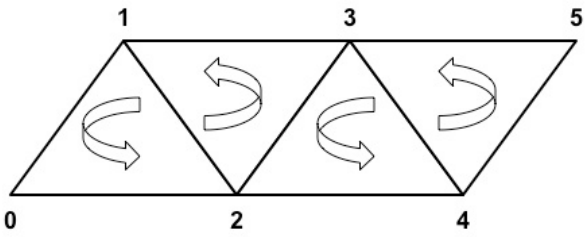

Figure 12: Tessellation of typical planar face

Examples of simple models, created in SolidEdge, are presented below. All models were exported to JT and then voxelized. Fig. 13 shows a peg, Fig. 14 shows a fixture with several circular through holes and Figure 15 shows a swept volume.

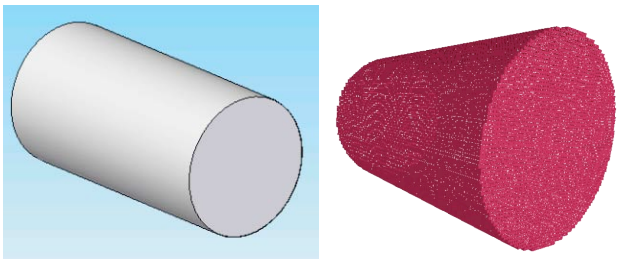

Figure 13: JT file in SolidEdge and voxelized model of a simple peg

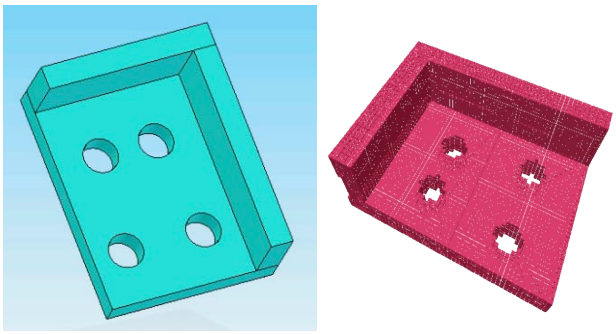

Figure 14: JT file in SolidEdge and voxelized model of a simple fixture with holes

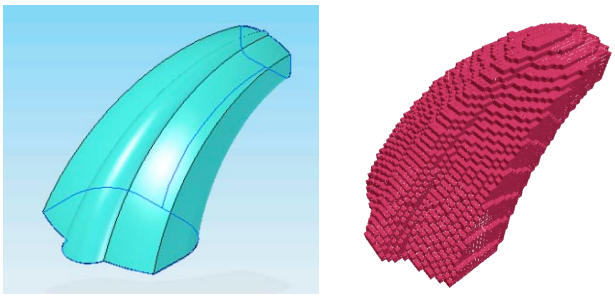

Figure 15: JT file in SolidEdge and voxelized model of a swept volume

Voxelization of the JT models are time-dependent on the voxel size, but occurs prior to the interactive simulation.

2. Tying BREP to Voxel data

VPS provides the capability to traverse the voxmap and query individual voxel information such as position. It is at this point that we propose to link the voxels to the BREP data.

The DCubed software family from Siemens UGS provides support for BREP to BREP collisions. Our approach is to create another Parasolid body (using the *parasolid_kernel.h* method), the size of a single voxel, and move it along the boundary of the voxelized model, moving between each specific voxel location. During the traversal of the voxelized model, each voxel can be accessed and its location is available. Using this positional information, the Parasolid block is moved accordingly. We perform a collision test between the Parasolid cube and the original CAD BREP to identify the BREP at that location. Details are described next.

After the voxelization process is complete, an instance of the DCubed Collision Detection is created. The BREP model is loaded using the D3E_BASE module. The D3E_BASE module holds the model data structure definitions and specifies which 3D geometry model format is used for the other D-Cubed components. A Parasolid cube, the same size as a voxel, is created in the initial step using the Parasolid call to *PK_BODY_create_solid_block* (Fig. 16).

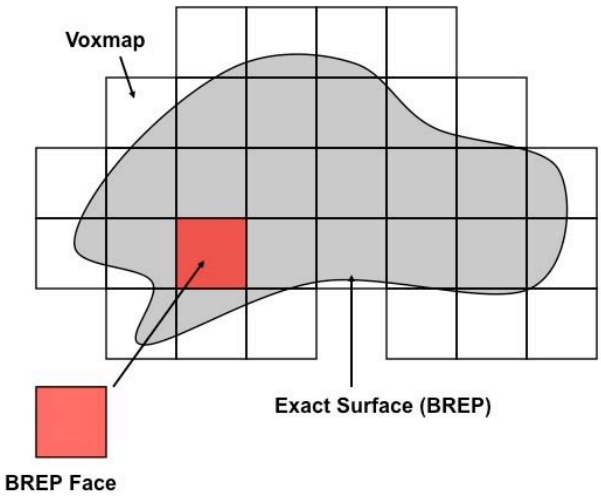

Figure 16: Association of BREP with voxels through collision detection (2D case)

The cube is moved to the exact location of a voxel based on the information from the voxel query. Next, a pair-wise collision detection occurs between the CAD model and the cube. DCubed's Collision Detection Manager (CDM) is responsible for managing collisions and interferences and uses the BREP data loaded by the D3E_BASE to query for collisions. Since the Parasolid cube is located at the exact location of the voxel, the BREP of the CAD model can be identified at that location and be associated with the individual voxel. BREP entities of the cube are ignored during this traversal.

The method checks all colliding faces and determines which type of face is currently in collision (i.e. cylindrical, planar, etc.). The method also checks what type of edge is colliding (i.e. circular, linear, elliptical, etc.). This information is saved onto each voxel. The voxel representation in VPS contains a data bin where private data can be stored.

A look up list is created for each voxel (Fig. 17). Any BREP data type can be saved into this look-up table. The face and edge BREP data is stored in the list using the function *VpsSetPrivateData*. The look-up table supports multiple BREPs. A pointer to the list is stored as part of the voxel data. This pointer has a unique identifier and allows access to the BREP on a voxel-by-voxel basis.

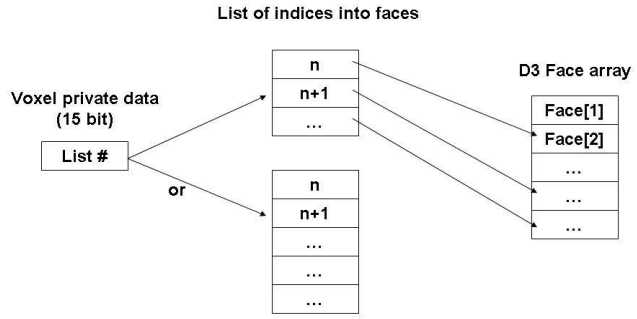

Figure 17: Look-up table

During the voxmap pointshell collision detection, the private data stored in each voxel can be accessed to reveal which faces or edges the voxels are colliding with. This tight link between voxels and BREPS supports future development of a hybrid approach to haptic interaction that is based on both voxel-based collision detection and force calculations and geometric constraint-based methods.

RESULTS

A unique way of associating BREPs with voxels has been developed using the CAD-neutral JT file format. Several objects were modeled in SolidEdge, converted to JT and then converted to a voxelized model. The BREP contained in the JT file was associated with each voxel. The advantage of combining voxel-based collision detection with BREP geometry is that exact geometry information is always available during collisions.

VR assembly has great potential to aid in the discovery of better designs for manufacturing and assembly. Allowing direct manipulation of parts has the potential to enhance collaboration between design and manufacturing team members. Providing better methodologies to handle complicated scenarios benefits the engineering design community directly, as well as other communities that desire low clearance object manipulation. Identification of boundary representations is critical to supporting low clearance assembly. Real-time implementations of such algorithms provide immediate feedback on the feasibility of assemblies. This paper presented a method to couple the boundary representations needed for low clearance assembly with the voxel-based models used in real time haptic rendering. Our next step is to explore using this model framework to support a hybrid method of virtual assembly.

ACKNOWLEDGMENTS

We are grateful for the technical assistance of William McNeely. The authors would like to thank the Iowa State University's Virtual Reality Applications Center for the use of computational resources and hardware. This research was funded by National Science Foundation grant CMMI - 0928774.

REFERENCES

[1] A. Gomes, and Zachmann, G. , "Virtual Reality as a Tool for Verification of Assembly and Maintenance Processes," *Computers and Graphics,* vol. 23, pp. 189-403, 1999.

[2] E. Pere, Langrana, N., Gomez, D., and Burdea, G., "Virtual Mechanical Assembly on a PC-Based System," in *ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC1996/DFM-1306)* Irvine, CA, 1996.

[3] R. Gupta and J. Krishnasamy, "Modeling and simulation of dynamic and haptic interactions in multimodal virtual environments," in *Proceedings of International conference on Virtual Systems and Multimedia*, Gifu, Japan, 1995, pp. 161-170.

[4] R. Gupta, T. Sheridan, and D. Whitney, "Experiments Using Multimodal Virtual Environments in Design for Assembly Analysis," *Presence,* vol. 6, pp. 318-338, 1997.

[5] R. Gupta, D. Whitney, and D. Zeltzer, "Prototyping and Design for Assembly Analysis using Multimodal Virtual Environments," *Computer Aided Design (Special issue on VR in CAD),* vol. 29, pp. pp. 585-597, 1997.

[6] A. S. Coutee, and, Bras, B., "Collision Detection for Virtual Objects in a Haptic Assembly and Disassembly Simulation Environment," in *ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2002/CIE-34385)* Montreal,Canada., 2002.

[7] A. S. Coutee, McDermott, S. D., and Bras, B., "A Haptic Assembly and Disassembly Simulation Environment and Associated Computational Load Optimization Techniques," *ASME Transactions - Journal of Computing & Information Science in Engineering,* vol. 1, pp. 113-122, 2001.

[8] C. E. Kim, and Vance, J.M., "Using VPS (Voxmap Pointshell) As The Basis For Interaction in a Virtual Assembly Environment," in *ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2003/CIE-48297)*, Chicago,IL., 2003.

[9] C. E. Kim, and Vance, J.M., "Development of a Networked Haptic Environment in VR to Facilitate Collaborative Design Using Voxmap Pointshell (VPS) Software," in *ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2004/CIE-57648)*
, Salt Lake City, UT., 2004.

[10] S. Jayaram, Jayaram, U., Wang, Y., Tirumali, H., Lyons, K. and, Hart, P., "VADE: A Virtual Assembly Design Environment," *Computer Graphics and Applications,* vol. 19, pp. 44-50, 1999.

[11] Y. Wang, Jayaram, S., Jayaram, U., and Lyons, K., "Physically Based Modeling in Virtual Assembly," in *ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2001/CIE-21259)*, Pittsburg, PA, 2001.

[12] X. Chen, N. Xu, and Y. Li, "A Virtual Environment for Collaborative Assembly," in *2nd International Conference on Embedded Software and Systems (ICESS'05)*, Xian, China, 2005.

[13] H. Wan, Gao, S., Peng, Q., Dai, G and Zhang, F., "MIVAS: A Multi-Modal Immersive Virtual Assembly System," in *ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC 2004/CIE-57660)*, Salt Lake City, UT., 2004.

[14] L. Tching, G. Dumont, and J. Perret, "Interactive Simulation of CAD Models Assemblies Using Virtual Constraint Guidance," *IJIDEM,* 2009.

[15] L. Marcelino, Murray, N., and, Fernando, T., "A Constraint Manager to Support Virtual Maintainability," *Computers & Graphics,* vol. 27, pp. 19 - 26, 2003.

[16] A. Seth, J. M. Vance, and J. H. Oliver, "Combining geometric constraints with physics modeling for virtual assembly using SHARP," in *ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Las Vegas, NV, September 4-7, 2007 (DETC2007-34681), 2007.

[17] R. Iacob, P. Mitrouchev, and J. Léon, "A Simulation Framework for Assembly/Disassembly Process Modeling," *Proceedings of the ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference,* 2007.

[18] R. Iacob, P. Mitrouchev, and J. Léon, "Contact identification for assembly–disassembly simulation with a haptic device," *The Visual Computer,* vol. 24, pp. 973-979, 2008.

[19] S. Jayaram, J. Vance, R. Gadh, U. Jayaram, and H. Srinivasan, "Assessment of VR Technology and its Applications to Engineering Problems," *Journal of Computing and Information Science in Engineering,* vol. 1, pp. 72-83, 2001.

[20] M. Lin, "Recent Advances in Haptic Rendering and Applications " pp. 1-312, Apr 22 2005.

[21] R. Wilson and J. Latombe, "Geometric reasoning about mechanical assembly," *Artificial Intelligence,* vol. 71, pp. 371-396, 1994.

[22] I. Aguinaga, D. Borro, and L. Matey, "Path-planning techniques for the simulation of disassembly tasks," *Assembly Automation,* vol. 27, pp. 207-214, 2007.

[23] W. McNeely, K. D. Puterbaugh, and J. J. Troy, "Voxel-Based 6-DOF Haptic Rendering Improvements," *Haptics-e,* vol. 3, 2006.

[24] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, "Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling.," in *26th Annual Conference on Computer Graphics and Interactive Techniques*, 1999.

[25] M. Wang and W. McNeely, "Quasi-Static Approximation for 6 Degrees-of-Freedom Haptic Rendering," in *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, Washington, DC, 2003, p. 34.

[26] C. E. Kim, and Vance, J.M., "Collision Detection and Part Interaction Modeling to Facilitate Immersive Virtual Assembly Methods," *ASME Journal of Computing and Information Sciences in Engineering,* vol. 4, pp. 83-90, 2004.
[27] M. Renz, C. Preusche, M. Poetke, H.-P. Kriegel, and G. Hirzinger, "Stable Haptic Interacton with Virtual Environments Using an Adapted Voxmap-PointShell Algorithm," in *Proceedings of the Eurohaptic Conference*, Birmingham, UK, 2001.
[28] M. Sagardia, T. Hulin, C. Preusche, and G. Hirzinger, "Improvements of the Voxmap-PointShell Algorithm–Fast generation of Haptic Data-Structures," *53rd Internationales Wissenschaftliches Kolloquium, Ilmenau,* 2008.
[29] D. Borro, A. Garcia-Alonso, and L. Matey, "Approximation of optimal voxel size for collision detection in maintainability simulations within massive virtual environments," *Computer Graphics Forum,* vol. 23, pp. 13-23, 2004.
[30] I. Stroud, *Boundary Representation Modelling Techniques*: Springer, 2006.